<!--Topicos Especiales y Avanzados-->

<!--Ing. Amy Diaz-->

Documentación de API {

<Por="Karen Lopez"/>

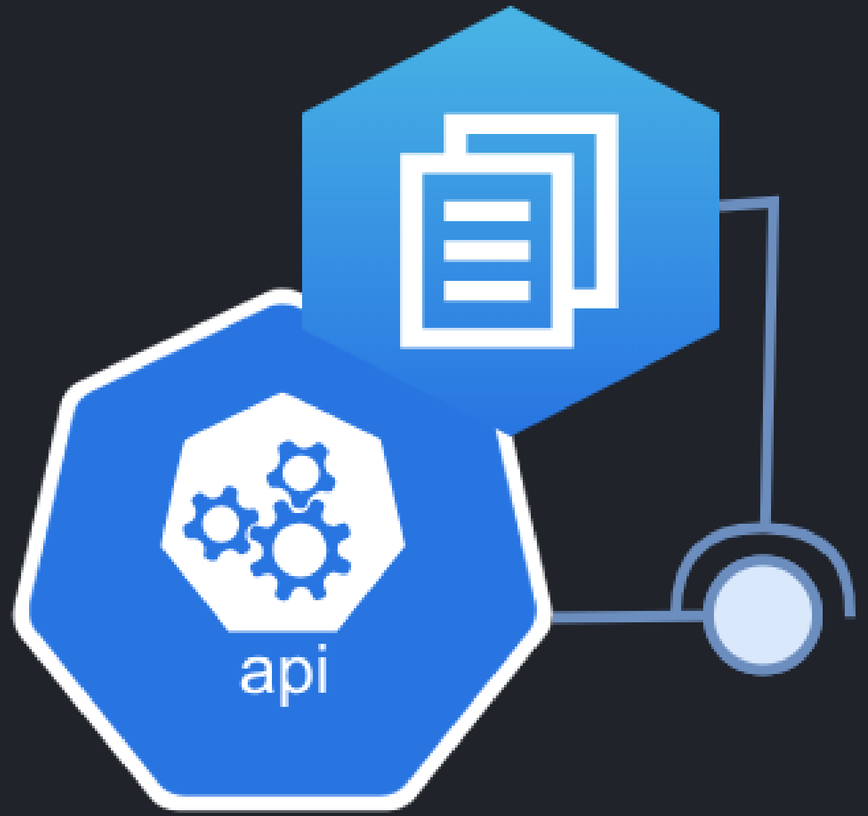
<Por="Junior Garcia"/>

}

Introducción {

¿Que es un API?

Interfaz de Programación de Aplicaciones



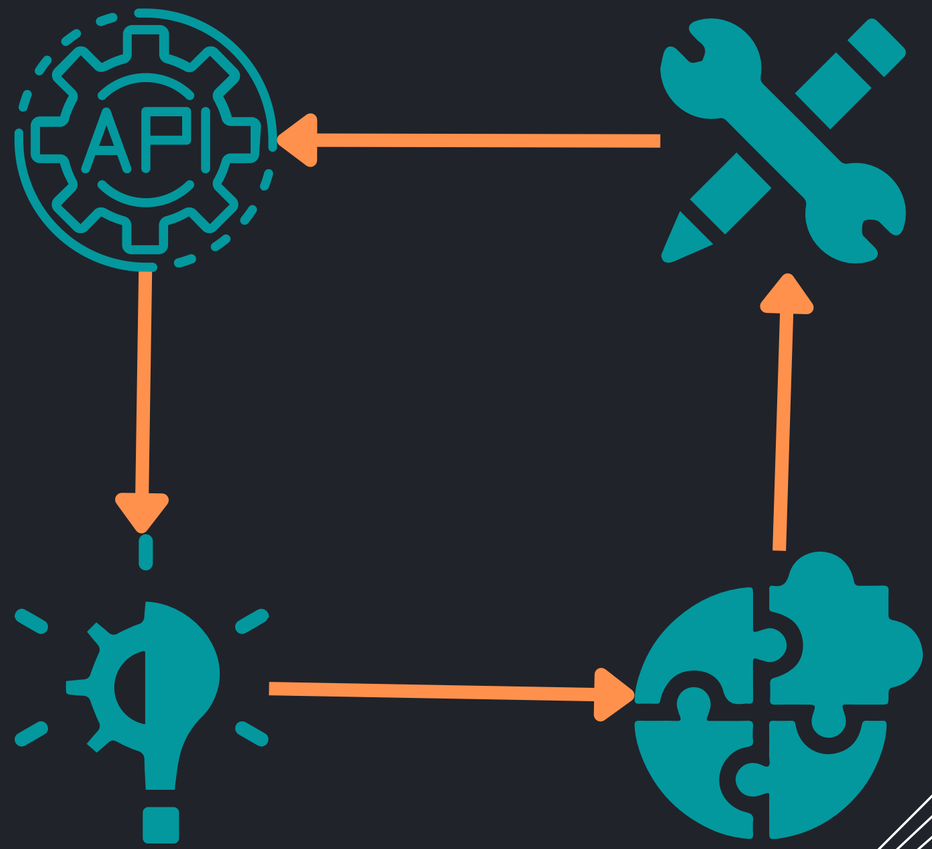
}

Beneficios y ejemplos de uso APIS

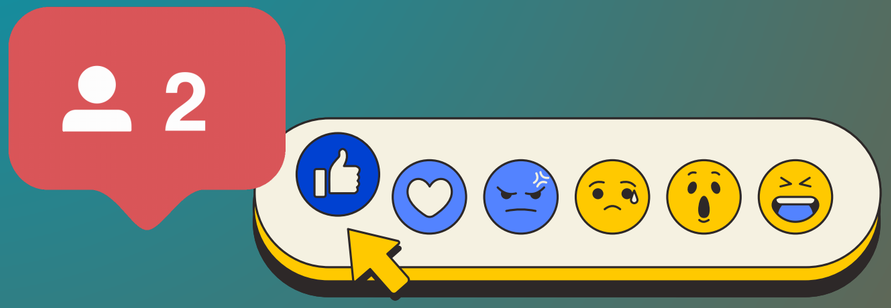
Beneficios de API REST {

- Integración
- Innovación
- Ampliación
- Facilidad de mantenimiento

}



Ejemplos de uso cotidiano

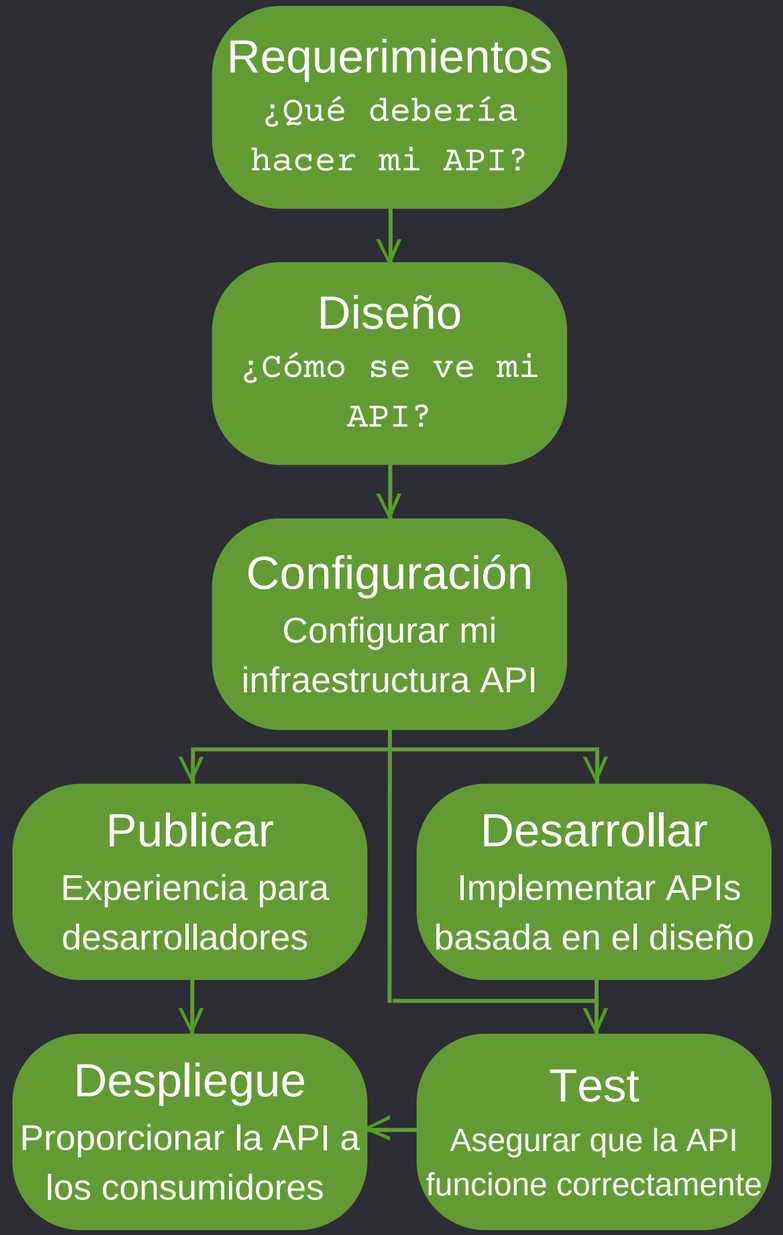


APIs de redes sociales

APIs de pasarelas de pago

APIs de servicios en la nube

Ciclo de vida de las APIs

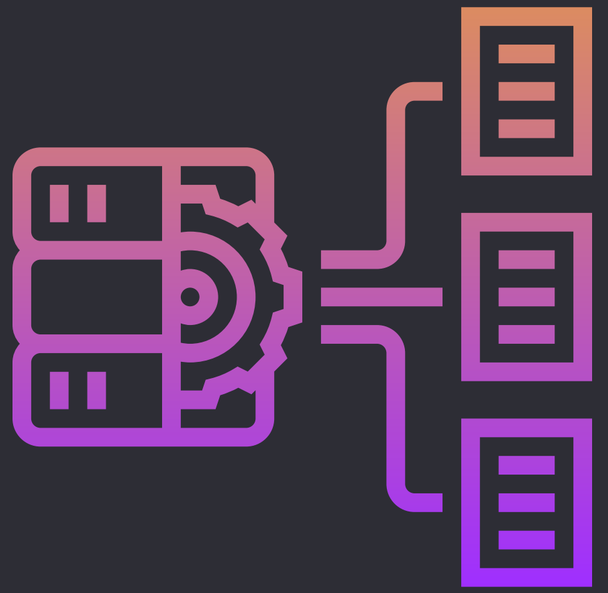


Tipos de Documentación de API

Documentación de referencia

Proporciona información técnica detallada sobre todos los endpoints, métodos, parámetros, tipos de datos, códigos de estado y cualquier otro aspecto técnico de la API.

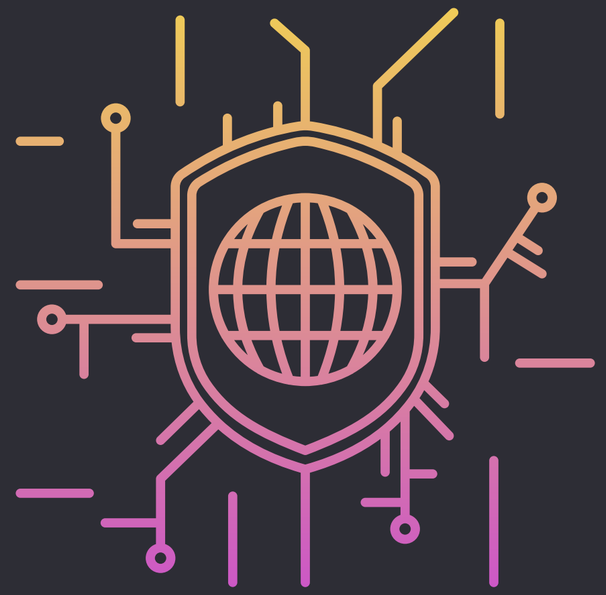
Ejemplos de documentación de referencia



Documentación de uso

Se centra en cómo utilizar la API de una manera más general. Incluye ejemplos y casos de uso que ayudan a los desarrolladores a comprender cómo aprovechar al máximo la API

Ejemplos de documentación de uso



Importancia de una Documentación

Si tu API no tiene documentación, nadie usará tu API en absoluto

Una buena documentación ahorrará tiempo en entender cómo funciona el API

Reducirá el número de preguntas innecesarias que tendría que responder el equipo de desarrollo

Errores de documentación

01 - No subirlos al repositorio

02 - No actualizar documentos/código

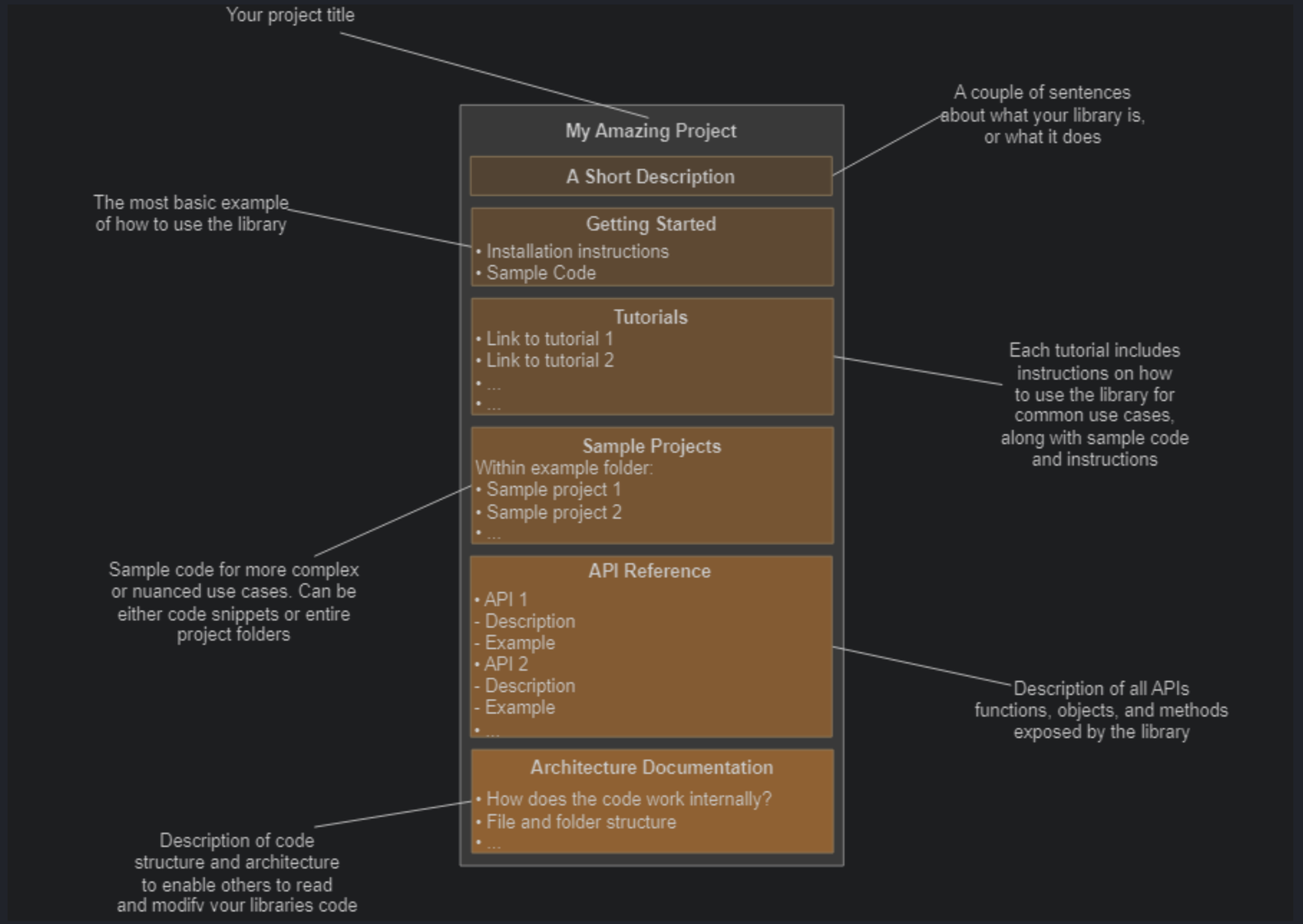
03 - No tener orden en el versionamiento

04 - No tener orden en las carpetas (arq, dev, qa)

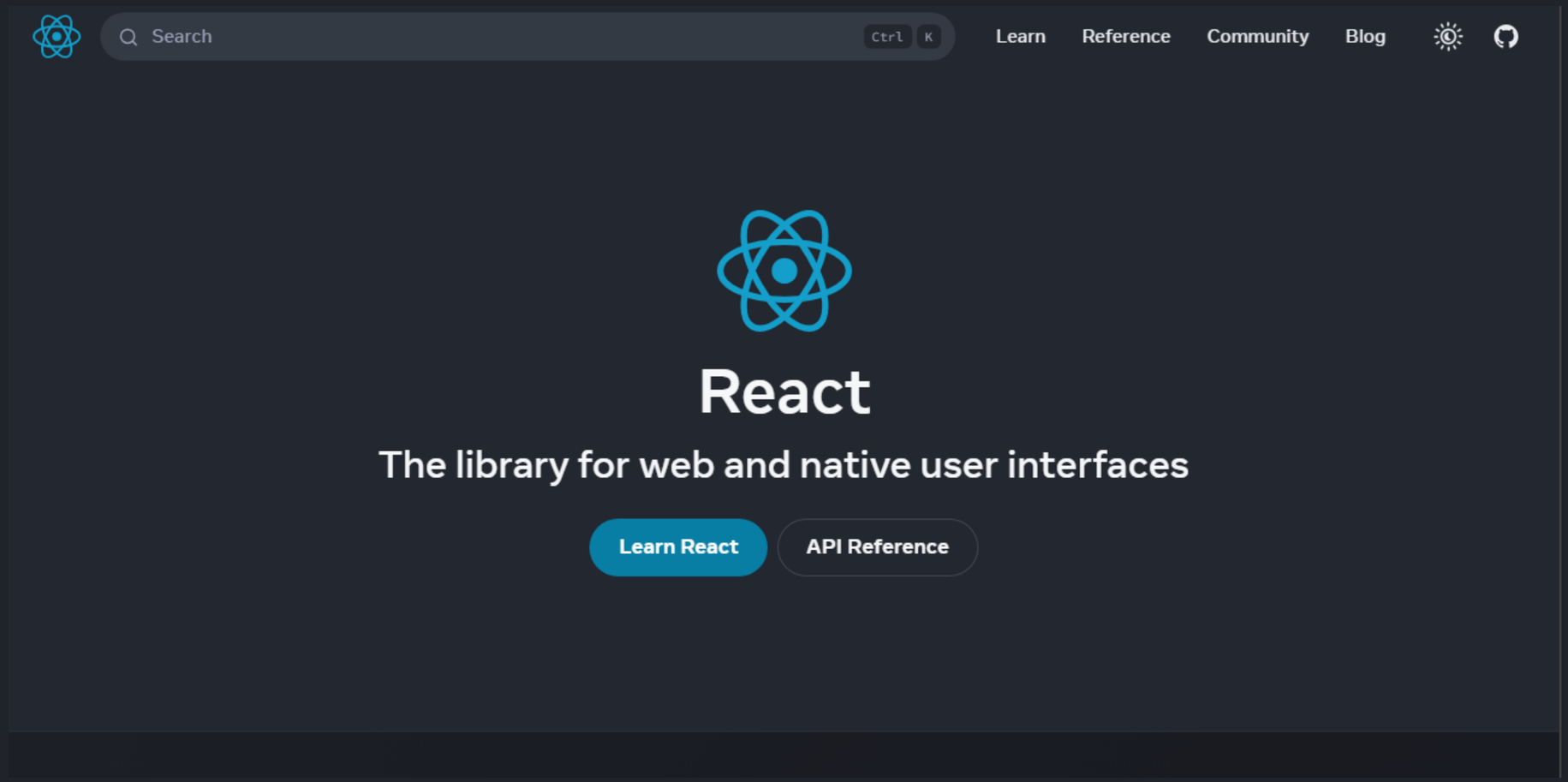
05 - No seguir el estandar (plantillas)

Elementos clave de una buena documentación

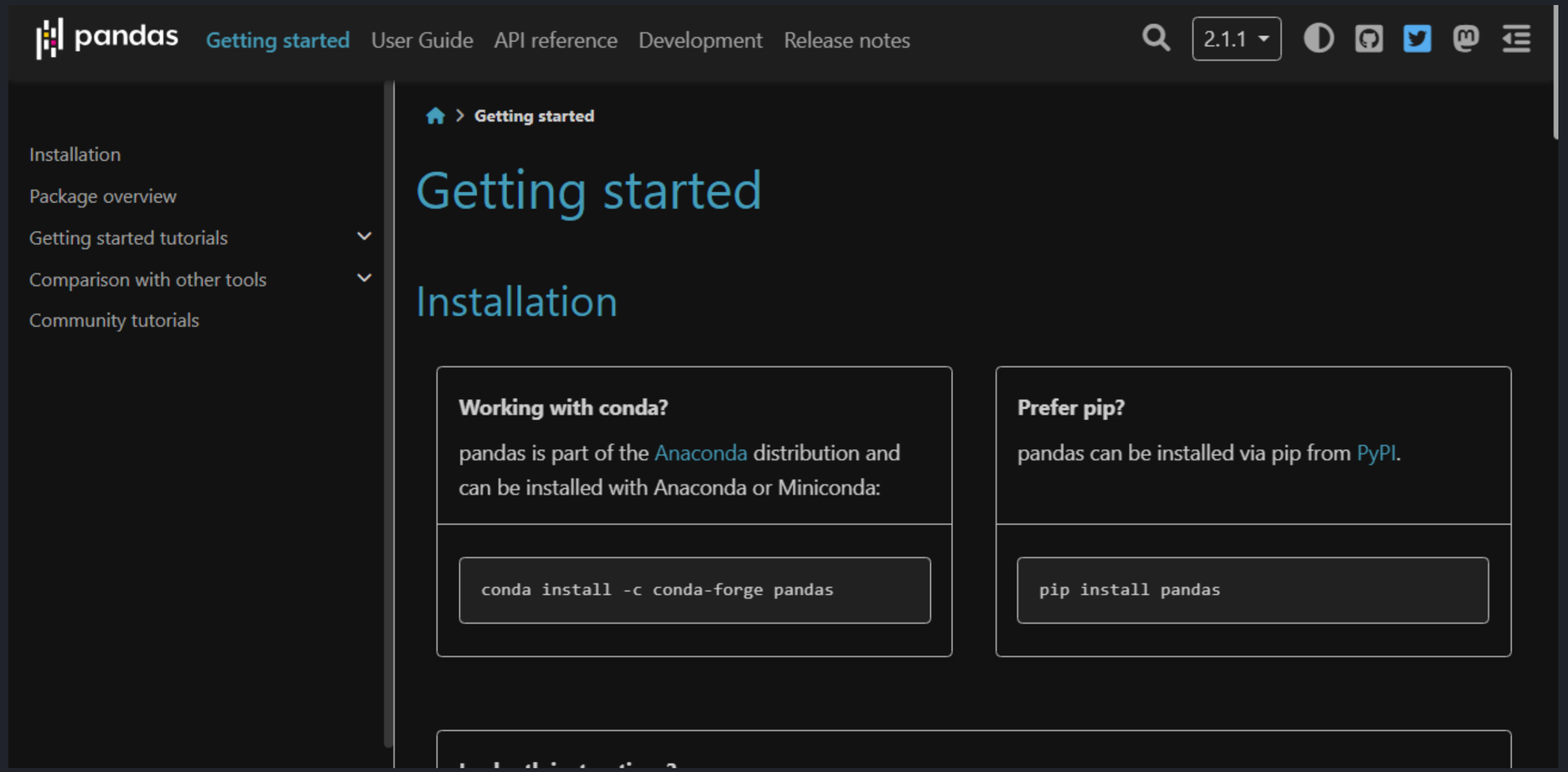
Elementos clave de una buena documentación {



Titulo del proyecto y breve descripción



Guia de inicio rapido



Tutoriales

GitHub, Inc. (US) | https://github.com/gin-gonic/gin#api-examples

API Examples

You can find a number of ready-to-run examples at Gin examples repository.

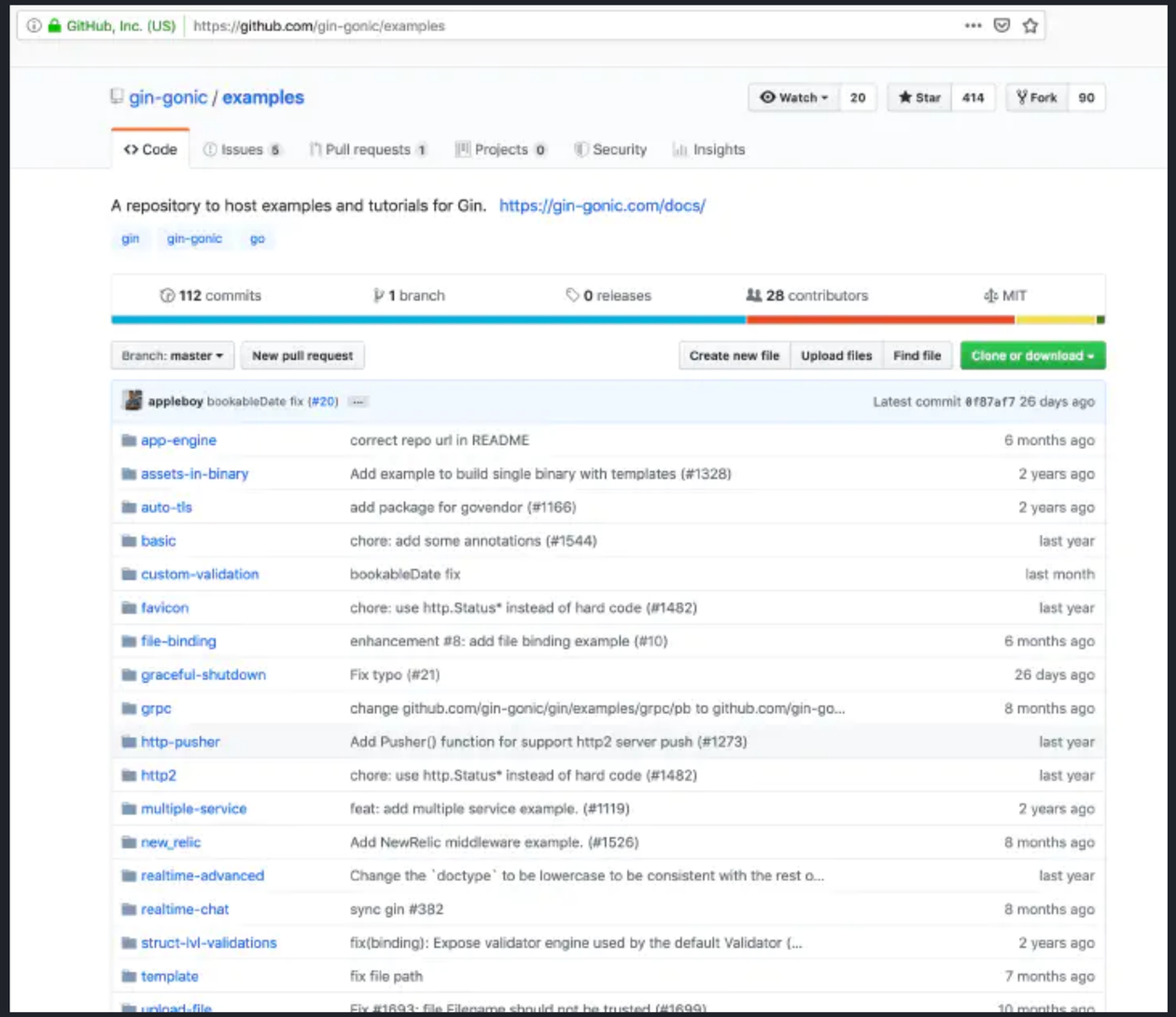
Using GET, POST, PUT, PATCH, DELETE and OPTIONS

```
func main() {
	// Creates a gin router with default middleware:
	// logger and recovery (crash-free) middleware
	router := gin.Default()

	router.GET("/someGet", getting)
	router.POST("/somePost", posting)
	router.PUT("/somePut", putting)
	router.DELETE("/someDelete", deleting)
	router.PATCH("/somePatch", patching)
	router.HEAD("/someHead", head)
	router.OPTIONS("/someOptions", options)

	// By default it serves on :8080 unless a
	// PORT environment variable was defined.
	router.Run()
	// router.Run(":3000") for a hard coded port
}
```

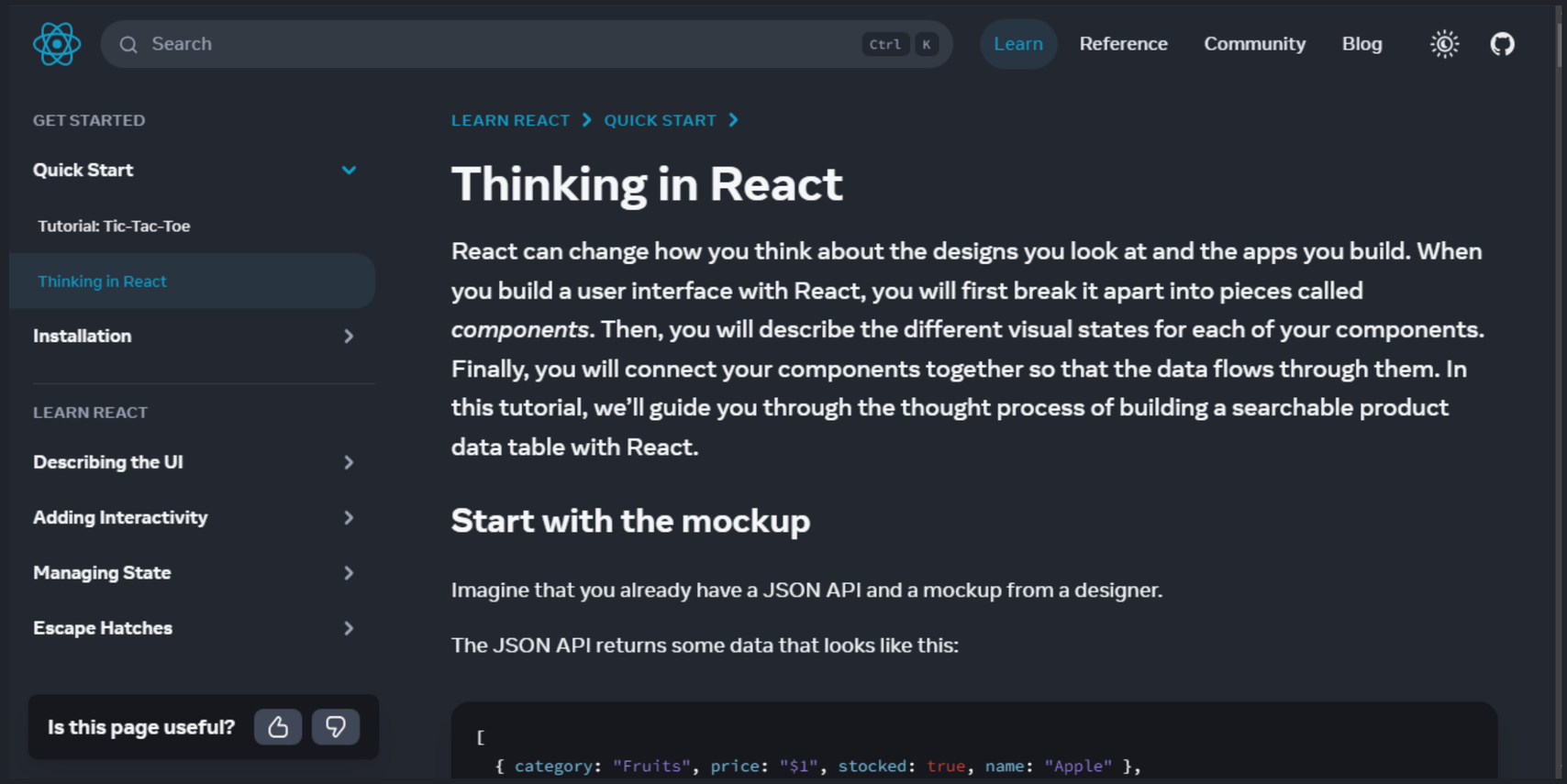
Parameters in path

```
func main() {
	router := gin.Default()

	// This handler will match /user/john but will not match /user/ or /user
	router.GET("/user/:name", func(c *gin.Context) {
		name := c.Param("name")
		c.String(http.StatusOK, "Hello %s", name)
	})

	// However, this one will match /user/john/ and also /user/john/send
	// If no other routers match /user/john, it will redirect to /user/john/
	router.GET("/user/:name/*action", func(c *gin.Context) {
		name := c.Param("name")
		action := c.Param("action")
```

Proyectos de muestra

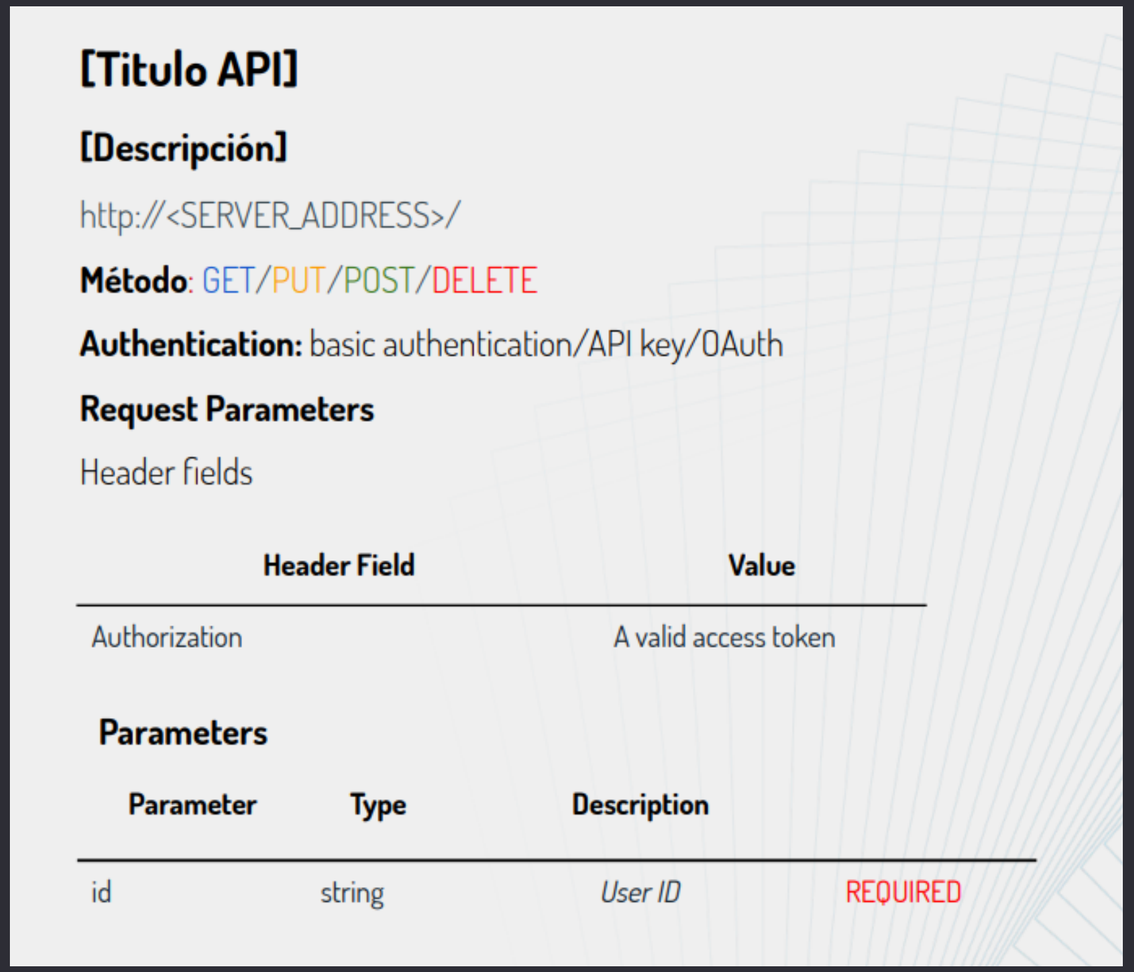


Documentación de arquitectura



Thinking in React

Rest API {



[Titulo API]

[Descripción]

http://<SERVER_ADDRESS>/

Método: GET/PUT/POST/DELETE

Authentication: basic authentication/API key/OAuth

Request Parameters

Header fields

Header Field	Value
Authorization	A valid access token

Parameters

Parameter	Type	Description	
id	string	*User ID*	REQUIRED

1 TITULO/INTRODUCCION

2 ENDPOINT

3 METODO

4 AUTENTICACION

5 PARAMETROS

JSON Request Body

Value	Type	Description	
serialNumber	string	*Serial Number*	REQUIRED

HTTP Response

HTTP Code	Description
200	Successful Response
401	Unauthorized

Error or Warning Response

invalid value for `field_name`

Unrecognized values for parameter paramname: value1, value2, value3

6 JSON REQUEST BODY

7 RESPONSE

8 EJEMPLO

Plantilla -> }

}

Mejores practicas en la Documentación

Mejores practicas en la Documentaciòn de API {

Planificar

Tener un plan de acción para elaborar la documentación

Actualizar la documentacion

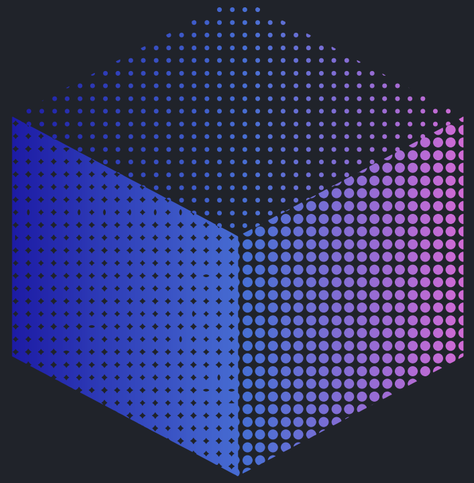
Asegurar que la documentación sea actualizada periódicamente

Responsable

Encargado de desarrollar y mantener la documentación

Pensar en la audiencia

Escribir la documentación pensando en la audiencia que la usara



Consistencia y evitar jerga

Documentación uniforme y con la misma terminología y nomenclatura

Crear ejemplos

Incluir código funcional y que pueda ser testeado de acuerdo a la documentación sugerida

Usar herramientas

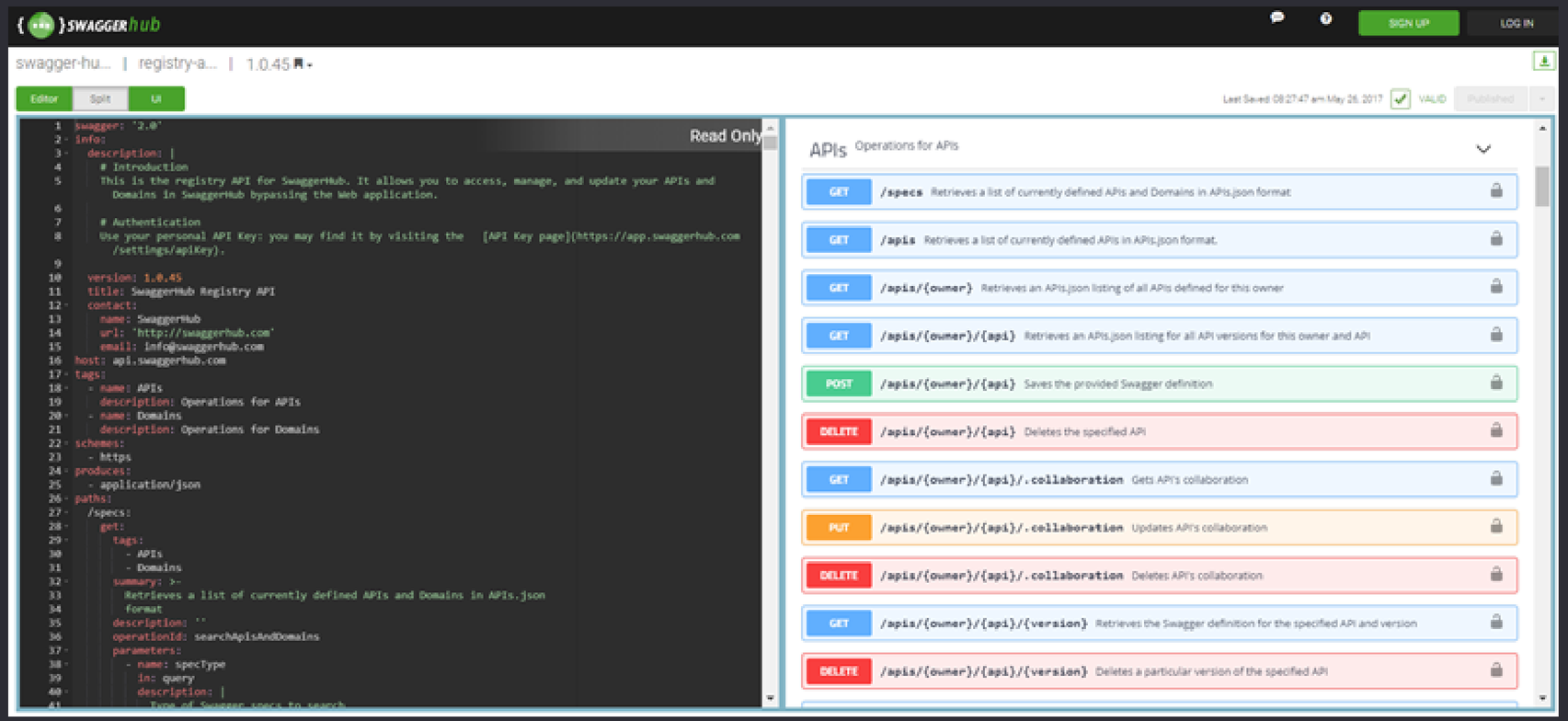
Permiten facilitar el trabajo y ayudan a crear la documentación de forma dinámica

}

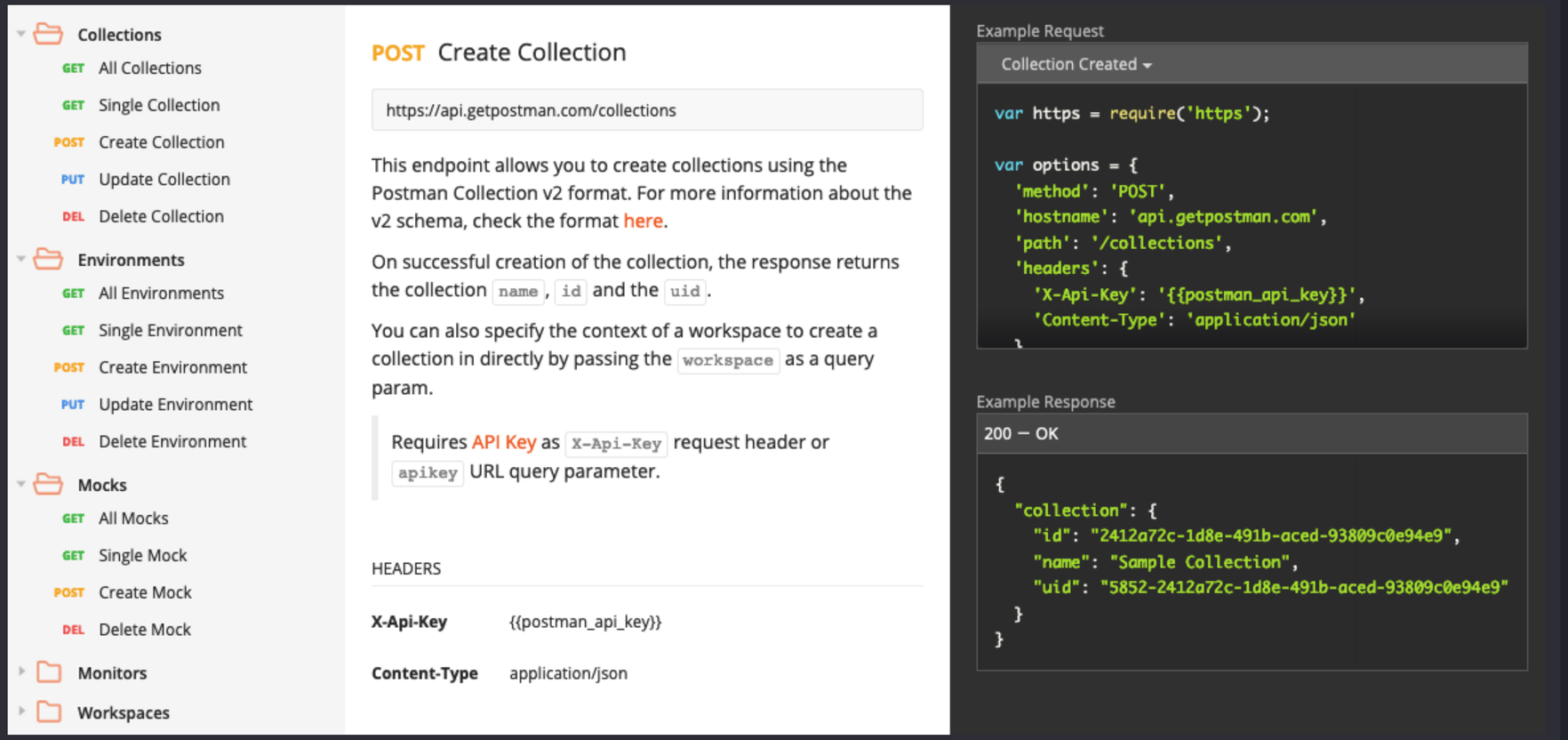
Herramientas de Documentación de API

HERRAMIENTAS{

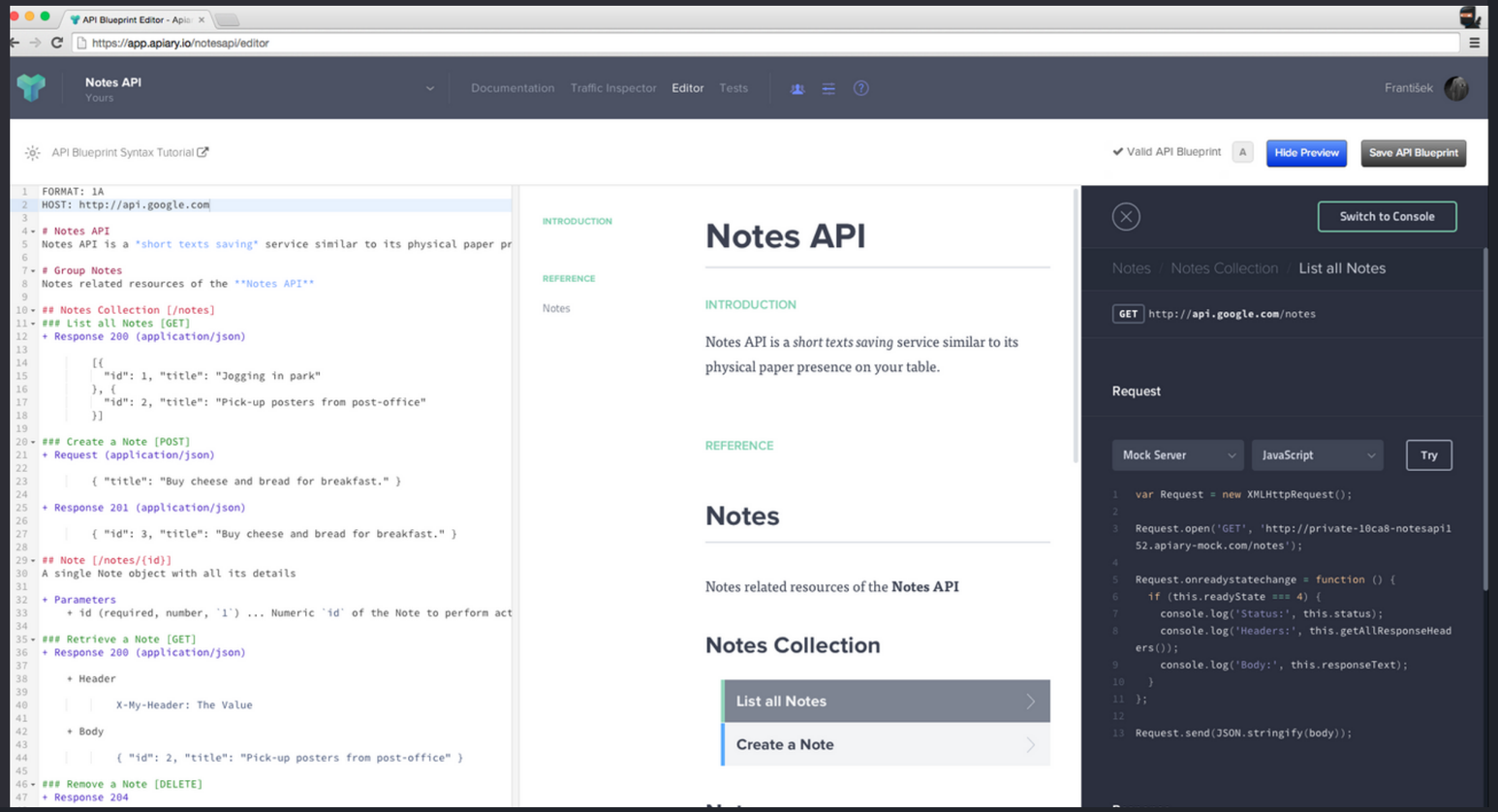
Swagger



Postman


Collections
GET All Collections
GET Single Collection
POST Create Collection
PUT Update Collection
DEL Delete Collection
Environments
GET All Environments
GET Single Environment
POST Create Environment
PUT Update Environment
DEL Delete Environment
Mocks
GET All Mocks
GET Single Mock
POST Create Mock
DEL Delete Mock
Monitors
Workspaces
POST Create Collection
https://api.getpostman.com/collections
This endpoint allows you to create collections using the Postman Collection v2 format. For more information about the v2 schema, check the format here.
On successful creation of the collection, the response returns the collection name, id and the uid.
You can also specify the context of a workspace to create a collection in directly by passing the workspace as a query param.
Requires API Key as X-Api-Key request header or apikey URL query parameter.
HEADERS
X-Api-Key {{postman_api_key}}
Content-Type application/json
Example Request
Collection Created
var https = require('https');
var options = {
'method': 'POST',
'hostname': 'api.getpostman.com',
'path': '/collections',
'headers': {
'X-Api-Key': '{{postman_api_key}}',
'Content-Type': 'application/json'
Example Response
200 — OK
{
"collection": {
"id": "2412a72c-1d8e-491b-aced-93809c0e94e9",
"name": "Sample Collection",
"uid": "5852-2412a72c-1d8e-491b-aced-93809c0e94e9"
}
}

Apiary


API Blueprint Editor - Apia
https://app.apiary.io/notesapi/editor
Notes API
Yours
Documentation
Traffic Inspector
Editor
Tests
František
API Blueprint Syntax Tutorial
Valid API Blueprint
A
Hide Preview
Save API Blueprint
FORMAT: 1A
HOST: http://api.google.com
Notes API
Notes API is a *short texts saving* service similar to its physical paper pr
Group Notes
Notes related resources of the **Notes API**
Notes Collection [/notes]
List all Notes [GET]
+ Response 200 (application/json)
[{
"id": 1, "title": "Jogging in park"
}, {
"id": 2, "title": "Pick-up posters from post-office"
}]
Create a Note [POST]
+ Request (application/json)
{ "title": "Buy cheese and bread for breakfast." }
+ Response 201 (application/json)
{ "id": 3, "title": "Buy cheese and bread for breakfast." }
Note [/notes/{id}]
A single Note object with all its details
+ Parameters
+ id (required, number, `1`) ... Numeric `id` of the Note to perform act
Retrieve a Note [GET]
+ Response 200 (application/json)
+ Header
X-My-Header: The Value
+ Body
{ "id": 2, "title": "Pick-up posters from post-office" }
Remove a Note [DELETE]
+ Response 204
INTRODUCTION
REFERENCE
Notes
Notes API
INTRODUCTION
Notes API is a short texts saving service similar to its physical paper presence on your table.
REFERENCE
Notes
Notes related resources of the Notes API
Notes Collection
List all Notes
Create a Note
Switch to Console
Notes / Notes Collection / List all Notes
GET http://api.google.com/notes
Request
Mock Server
JavaScript
Try
var Request = new XMLHttpRequest();
Request.open('GET', 'http://private-10ca8-notesapi152.apiary-mock.com/notes');
Request.onreadystatechange = function () {
if (this.readyState === 4) {
console.log('Status:', this.status);
console.log('Headers:', this.getAllResponseHeaders());
console.log('Body:', this.responseText);
}
};
Request.send(JSON.stringify(body));

}

Ejemplo de Documentación de API


{...}
Swagger™

Entidades que utilizan OPENAPI

ebay

IBM

Rapid

Microsoft

SAP

apiplatform.io

IFS

Geek.Zone

cisco

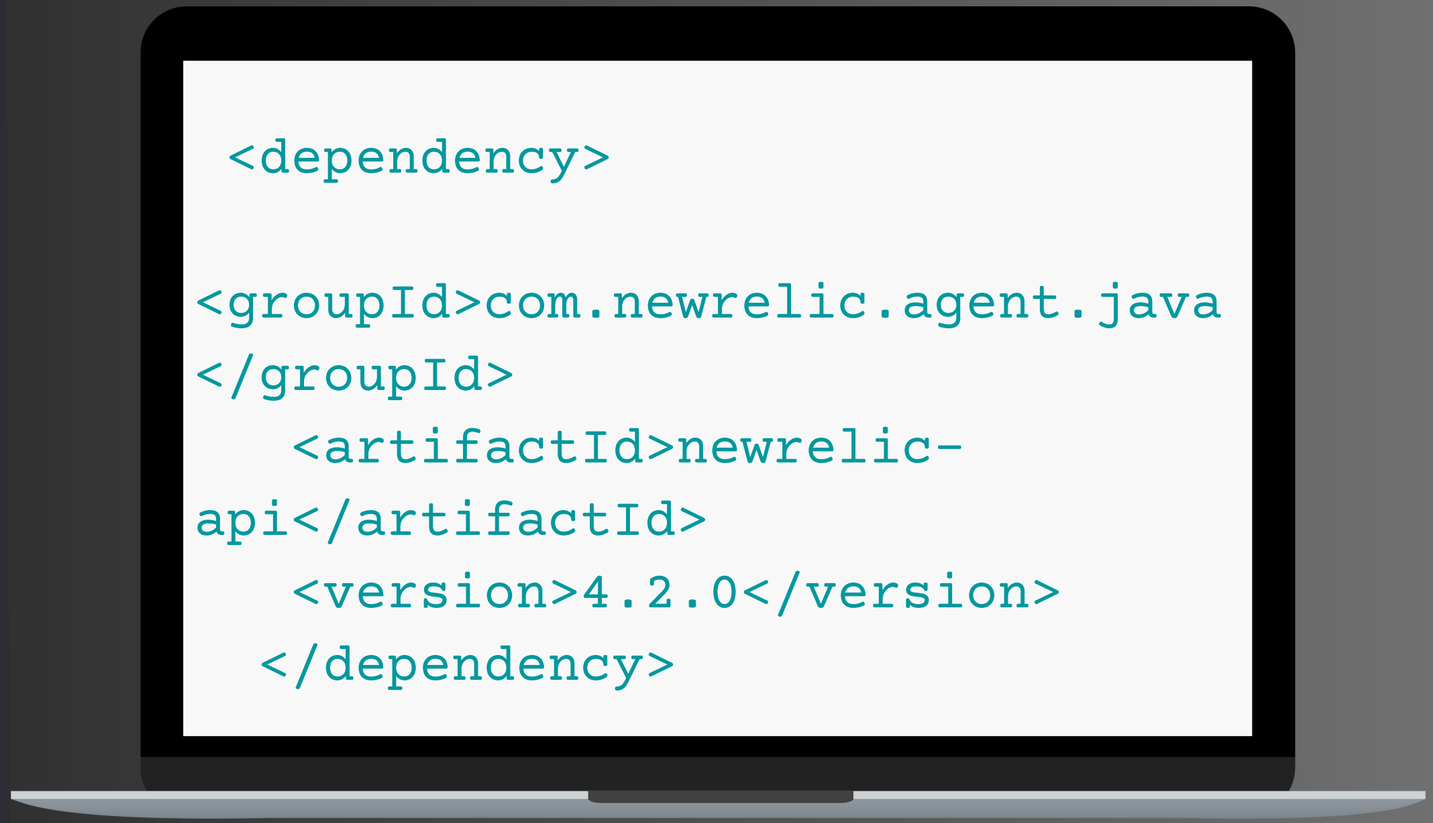
Google

SWAGGER {

Es un estándar que describe, produce, consume y visualiza REST API, APIs web services.

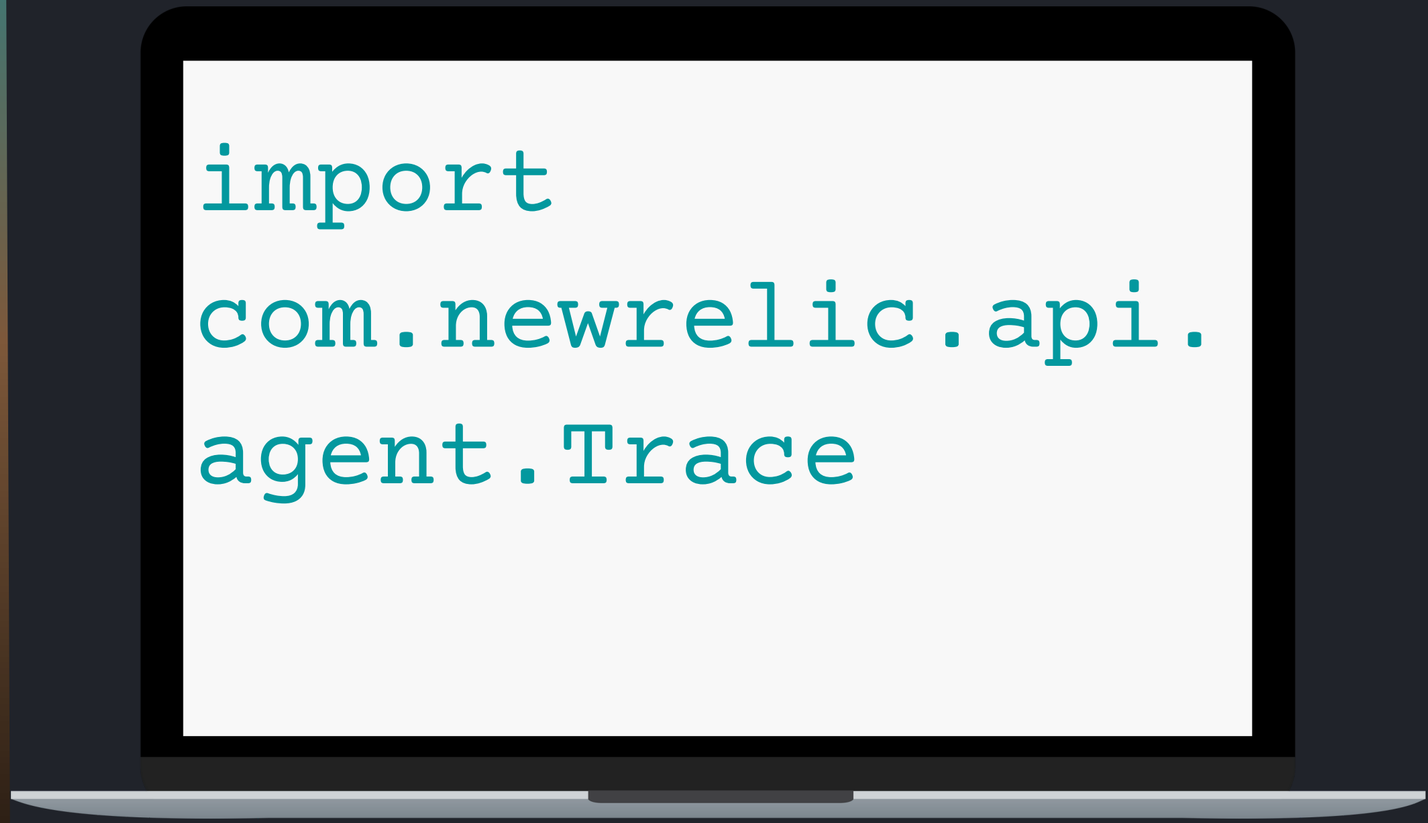
PASOS{

Agregar dependencia en el pom.xml

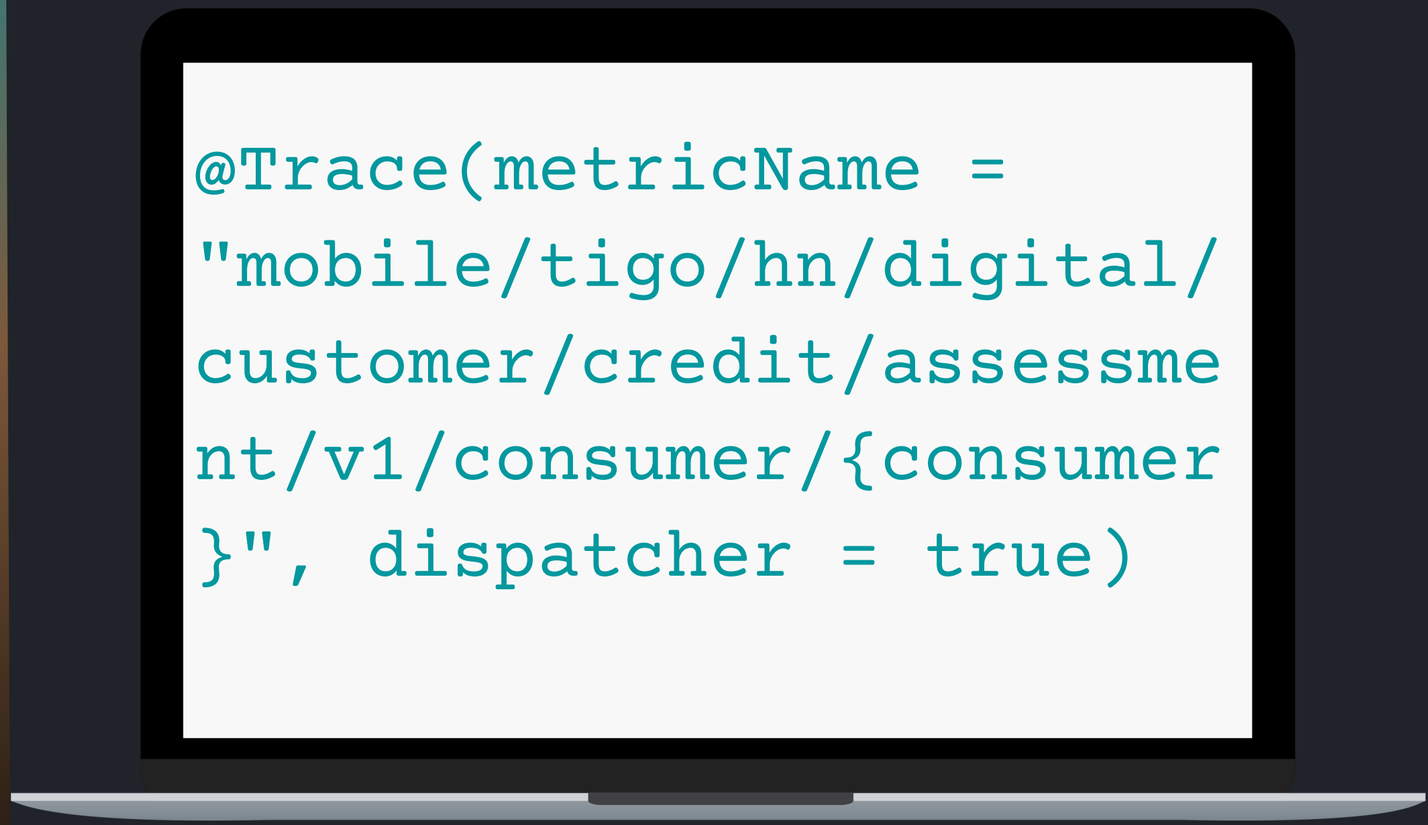


PASOS{

Agregar los import



En el método getrequest agregamos la URL de nuestra API:



Vamos a SwaggerEditor e importamos nuestro archivo :

}

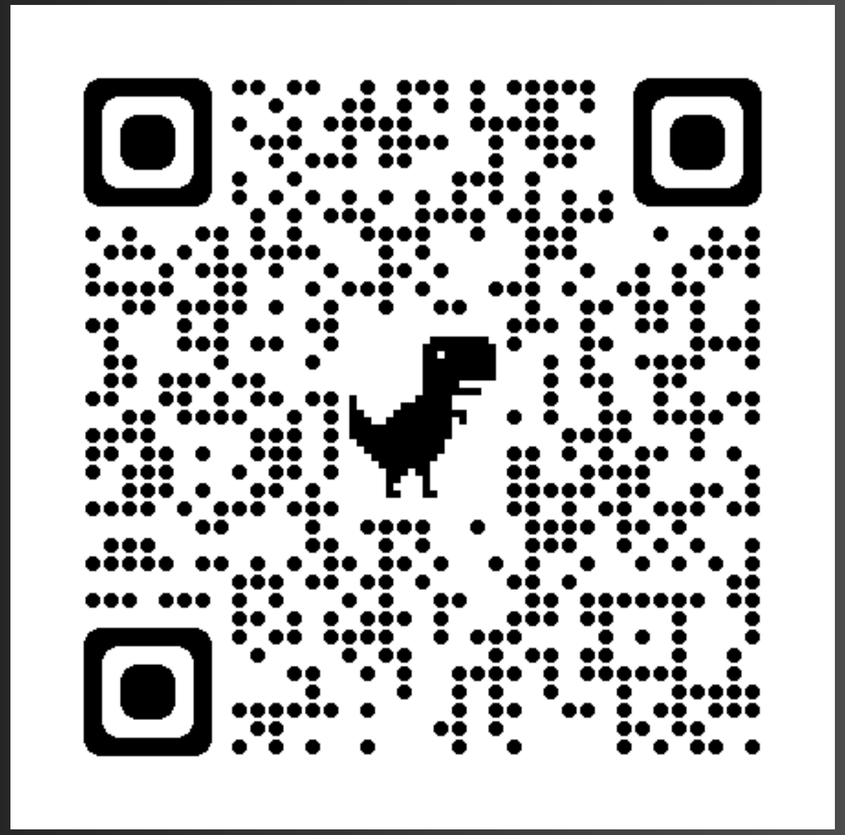
Conclusiones {

"La documentación es una carta de amor que le escribes a tu yo futuro". --Damian Conway

}

Enlace de Presentacion

{



Documentación

}

<!--Grupo # 14 -->

Gracias {

<Por="tu atención">

}